VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GENERÁTOR ZMĚN OBRAZU BURZY

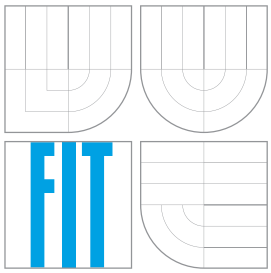
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ CIENCIALA

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GENERÁTOR ZMĚN OBRAZU BURZY

ORDER BOOK UPDATES GENERATOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ CIENCIALA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MILAN DVOŘÁK

BRNO 2014

Abstrakt

Tento dokument analyzuje zprávy zasílané burzami NYSE Arca a ISE a popisuje návrh generátoru, který generuje zprávy měnící obraz burzy. Lze ho využít pro testování programů, které pracují s informacemi zasílanými elektronickými burzami. Jsou popsány techniky verifikace řízené pokrytím a generování náhodných vstupních vektorů. Generování zpráv je založeno na XML šabloně, díky čemuž může být generátor použit pro různé burzy.

Abstract

This thesis analyses messages that come from NYSE Arca and ISE exchanges and provides a description of design of stock exchange updates generator which is capable of generating constrained-random messages. It can be used for testing software that handles messages from an electronic stock exchange. Techniques of coverage-driven verification and constrained-random stimulus generation are discussed. Message generation is based on XML template and because of that the generator can be adjusted for various exchanges.

Klíčová slova

generátor změn obrazu burzy, NYSE Arca, ISE, funkční verifikace, náhodné vstupní vektory, verifikace řízená pokrytím

Keywords

exchange updates generator, NYSE, ISE, functional verification, constrained-random messages, coverage-driven verification

Citace

Ondřej Cienciala: Generátor změn obrazu burzy, bakalářská práce, Brno, FIT VUT v Brně, 2014

Generátor změn obrazu burzy

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Milana Dvořáka a uvedl jsem všechny použité zdroje, ze kterých jsem čerpal.

.....
Ondřej Cienciala
31. července 2014

Poděkování

Chtěl bych poděkovat panu Ing. Milanovi Dvořákovi za vedení, přínosné konzultace a veškerý čas, který mi věnoval.

Obsah

1 Úvod **3**

2 Finanční burzy **4**
2.1 Znaky a funkce burzy 4
2.2 Historie burz 5
2.3 Elektronické obchodování na burzách 5
2.4 Algoritmické obchodování 6
2.5 Hloubka trhu 6

3 Komunikační protokol **8**
3.1 Formát zpráv 8
3.2 Variabilita zpráv 9

4 Funkční verifikace **10**
4.1 Možnosti verifikace 10
4.2 Generování náhodných vstupních vektorů 11
4.3 Verifikace řízená pokrytím 11

5 Šablona pro generování **13**
5.1 Požadavky na návrh šablony 13
5.2 Popis XML šablony 14

6 Volba generátoru náhodných čísel **16**
6.1 Generátor pravých náhodných čísel 16
6.2 Generátor pseudonáhodných čísel 17
6.3 PRNG v jazyce Python 17

7 Implementace aplikace Generátor **19**
7.1 Volba programovacího jazyka 19
7.2 Struktura programu 19
7.3 Parametry programu 21
7.4 Implementace metod funkční verifikace 22

8 Experimenty s nastavením Generátoru **24**
8.1 Malý stavový prostor 24
8.2 Středně velký stavový prostor 25
8.3 Shrnutí 26

9 Analýza reálných dat z burzy **28**
9.1 Vzorky dat 28
9.2 Vzorek č. 1 28
9.3 Vzorek č. 2 29
9.4 Vzorek č. 3 29
9.5 Vzorek č. 4 30
9.6 Shrnutí 30

10 Závěr **31**
10.1 Další možnosti práce 31

A Obsah CD **34**

Kapitola 1

Úvod

Vznik elektronických finančních burz přinesl nové způsoby obchodování. Především algoritmické obchodování prošlo v posledních letech velkým rozvojem. Díky tomu vznikají různé programy a systémy, které samy obchodují nebo alespoň usnadňují obchodníkům jejich činnost. To klade vysoké požadavky na bezchybovost těchto programů.

Cílem této práce bylo navrhnout a implementovat aplikaci, která dokáže generovat velké množství náhodných zpráv (ovšem vyhovujícím burzovním specifikacím), které může elektronická burza zasílat svým klientům. Toto umožní částečně simulovat proud dat, který přichází z burzy, a především testovat chování programů, které zprávy z burzy zpracovávají. Testování těchto konkrétních programů ale již není účelem této práce.

Při analýze a implementaci jsem se soustředil na burzy NYSE Arca (New York Stock Exchange) a ISE (International Securities Exchange). Ze zpráv, které tyto burzy zasílají, se zaměřuji na ty, které aktualizují hloubku trhu.

V kapitole 2 popisuji, jak fungují burzy a elektronické obchodování na burzách a co je to hloubka trhu (obraz burzy). V kapitole 3 rozebírám zprávy z burzy NYSE Arca a ISE. Ve 4. kapitole popisuji, jakým způsobem by bylo možné využít některých metod funkční verifikace pro účely této práce. V kapitole 5 definuji požadavky na šablonu, na základě které by mohlo probíhat generování zpráv, a následně tuto šablonu implementuji a popisuji. V kapitole 6 se pozastavuji u generátorů náhodných čísel a implementaci samotného programu rozvádím v kapitole 7. Experimenty s různým nastavením programu popisuji v kapitole 8 a v kapitole 9 se zabývám analýzou reálných dat z burzy. Závěru je věnována kapitola 10.

Kapitola 2

Finanční burzy

V této kapitole popisuji burzy a jejich fungování a také elektronické obchodování. Vycházel jsem ze zdrojů [10], [13] a [2].

2.1 Znaky a funkce burzy

V tržní ekonomice mají dobře fungující burzy velmi významnou úlohu. Střetávají se zde totiž ti, kteří nabízejí kapitál, s těmi, jenž jej naopak potřebují. To umožňuje orientovat kapitál tam, kde jej lze co nejúčelněji použít.

Jednou z důležitých funkcí burzy je stanovení kurzu. Díky intenzivnímu střetu nabídky s poptávkou, k jakému zde dochází, se projevuje jejich skutečný poměr, a to se promítne v kurzu. Pohyby kurzů jsou zajímavým ukazatelem, protože se do nich promítá současný i očekávaný vývoj obchodovaných předmětů, hospodářská i politická situace a mnoho dalších věcí.

Samotným pojmem burza označujeme vysoce organizovanou formu trhu, která má na rozdíl od jiných druhů trhů jisté specifické rysy. Důležité je především to, že prodávané a kupované předměty (např. akcie, nerostné suroviny či zemědělské plodiny) nejsou na trhu vůbec přítomny, ale obchodují se pomocí zástupných předmětů. Ty jsou určitým způsobem standardizovány nebo normalizovány, aby je šlo navzájem zaměnit mezi sebou. Vymezují se také minimální obchodovatelná množství, způsob stanovení cen, způsob placení atd. Díky tomu je obchodování jednodušší a transparentnější.

Dalším rysem burzy je pravidelnost obchodování na určitém místě. Aby se obchodníci navzájem neminuli, musí mít jistotu, že se obchoduje ve stanovaných dnech a hodinách. Proto má každá burza své vlastní burzovní hodiny, které jsou pečlivě dodržovány a během kterých je možno obchodovat.

Aby mohl být nějaký trh označen za burzu, musí být také přesně vymezen okruh osob, které jsou zde oprávněny obchodovat. O těchto podmínkách rozhodují orgány burzy. Například burza New York Stock Exchange umožňuje účast pouze svým členům, což mohou být jen fyzické osoby. Pokud chce nový člen získat možnost zde obchodovat, musí si pronajmout nebo koupit tzv. křeslo od jiného člena burzy. Počet křesel je již dlouhou dobu neměnný a činí 1 366.

Na burze chce samozřejmě působit velké množství různých subjektů a počet křesel pro všechny nestačí. Proto část členů burzy funguje jako tzv. brokeři. To jsou zprostředkovatelé, kteří uzavírají obchody pouze na cizí účet jménem jiných subjektů.

2.2 Historie burz

První známky něčeho, co by se dalo nazvat předchůdcem burzy, pochází již ze starověku, kdy Římané zakládali kupecká kolegia (collegia mercatorum). Později ve 12. století se objevují burzy zboží a peněžní burzy v severoitalských městech, kde se obchodníci a bankéři scházeli při příležitosti tradičních trhů.

Nejčastěji se za vznik burzy pokládá rok 1409, kdy v belgickém městě Bruggy žila rodina bankéře Van Buerse. Obchodníci se shromažďovali před jeho domem a jméno tohoto bankéře možná dalo vzniknout slovu burza. Později je v Antverpách otevřena burza pro obchod se směnkami a zlatými a stříbrnými mincemi. V 15. a 16. století dále vznikají burzy ve významných městech západní Evropy a jsou stále více a více organizované. Staví se burzovní budovy, kam mají přístup pouze členové, zavádí se různá pravidla obchodování a burzy se postupně stávají více specializované (např. burzy cenných papírů, plodinové burzy).

Technický pokrok především ve druhé polovině 19. století přináší nové nástroje pro šíření informací a domlouvání obchodů. Využívá se potrubní pošta, telegraf a později především telefon. Největší změny přichází s rozvojem výpočetní techniky, kdy se pro zadávání obchodních příkazů začínají využívat elektronické systémy a počítačové sítě. Ke konci 20. století se začíná rozmáhat elektronické obchodování na některých burzách, kde účastníci mohou v přímém přenosu sledovat aktuální dění na burze a okamžitě na ně reagovat. To dává vzniknout jednak novým finančním instrumentům, tak i novým typům elektronického obchodování za pomoci počítačů.

Obrázek 2.1: Klasické obchodování na parketu - burza NYSE (2013). Převzato z http://www.ibtimes.com/.

2.3 Elektronické obchodování na burzách

Na rozdíl od klasického původního typu obchodování, které se realizuje fyzicky v prostorách burzy nebo obchodování přes telefon, přináší elektronické obchodování řadu výhod. Snižuje počáteční náklady pro obchodníky a cenu transakcí či zvyšuje likviditu. Účastníci elektronické burzy se střetávají online a realizují své obchody přes elektronický obchodovací systém.

Obchodníci můžou zadávat příkazy sami nebo mohou využívat programy, které díky

pokročilým algoritmům mohou obchodovat samostatně.

2.4 Algoritmické obchodování

Odhaduje se, že algoritmické obchodování, resp. jeho speciální typ označovaný jako vysokofrekvenční obchodování, dnes tvoří většinu objemu transakcí, které jsou uskutečněny v rámci Spojených států a Evropské unie. Algoritmické obchodování funguje na pricipu, kdy obchodní příkazy jsou vytvářeny specializovanými programy a většinou jsou vykonávány bez zásahů člověka. Využívají ho nejrůznější finanční subjekty jako například banky, fondy, tvůrci trhu či jednotliví obchodníci a investoři.

Vysokofrekvenční obchodování

Vysokofrekvenční obchodování označované zkratkou HFT[1] používá strategii velmi rychlých nákupů a prodejů v řádech zlomků sekundy. Program pomocí různých algoritmů vyhodnocuje pohyby cen, velikosti objemu příkazů na různých cenových hladinách a další komplexnější informace. Obchody se realizují ve velkém množství a většinou s velmi nízkým ziskem na jednotlivém obchodu. Ukazuje se také, že spíše než používaní složitých rozhodovacích algoritmů může být výhodnější provést obchod na základě rozhodnutí, které zohledňuje sice méně různých informací, ale je vykonané rychleji a předběhne konkurenci.

Software, který provádí HFT či jakékoliv algoritmické obchodování je velice důležité před nasazením testovat. Neodhalené chyby zde můžou za velmi krátkou dobu způsobit katastrofální finanční ztráty. Nejznámější je případ amerického brokera a tvůrce trhu Knight Capital Group, který v roce 2012 vlivem nasazení chybného programu na produkční server začal vytvářet enormní množství nevýhodných obchodních příkazů na NYSE burze. Každou minutou způsobil ztrátu 10 milionů dolarů. Program byl po 45 minutách vypnut a celková ztráta dosáhla přibližně 440 milionů dolarů.

2.5 Hloubka trhu

Hloubka trhu je ukazatel, který nás informuje o tom, jaká je okamžitá nabídka a poptávka na různých úrovních kurzu. Obchodníci zadávají příkazy k nákupu či k prodeji, které se mají vykonat, pokud cena buď klesne, nebo stoupne přes určitou mez. Nebo se dá také říct, že hloubka trhu je velikost obchodu, který musí být proveden, aby se kurz pohnul na určitou úroveň. Analýza těchto dat pomáhá obchodníkovi nebo programu činit obchodní rozhodnutí (koupit nebo prodat).

Hloubka trhu neboli obraz burzy je obvykle zobrazen uživateli formou tabulky s několika nejlepšími nabídkami a poptávkami. Může mít také podoby různých grafů.

Obrázky 2.2 a 2.3 ukazují některé z možných forem grafického znázornění hloubky trhu.

Konkrétně u burzy NYSE Arca i ISE má hloubka trhu podobu tabulky o pěti nejlepších cenách pro stranu nabídky a pro stranu poptávky a ke každé ceně korespondující velikost obchodního příkazu, který by musel být proveden, aby bylo této ceny dosaženo.

[1] HFT - High-Frequency Trading

Share Quote as at **1:06 PM Sydney Time, Tuesday, 31 July 2007**

▲ WOODSIDE PETROLEUM FPO ▲ CommSec Margin Lending LVR: 70%

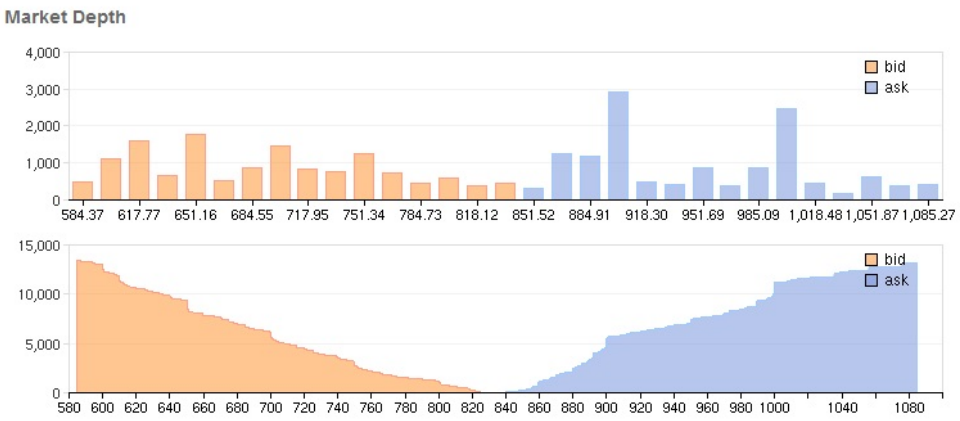
Code	Bid	Offer	Last	Change'	Open	High	Low	Volume	News
WPL	43.130	43.140	43.140	+0.390	43.150	43.210	42.960	912,917	

Buy | Sell | Add to Watchlist | Research | Chart | Print | Help

Market Depth

BUY				SELL		
Number	Quantity	Price	#	Price	Quantity	Number
4	1,428	43.130	1	43.140	1,152	4
1	587	43.120	2	43.150	1,414	1
1	225	43.110	3	43.180	2,220	5
3	4,097	43.100	4	43.190	6,857	3
1	291	43.070	5	43.200	10,384	3
3	1,623	43.060	6	43.220	1,100	1
2	3,844	43.050	7	43.230	1,500	1
1	958	43.030	8	43.240	476	1
1	1,747	43.020	9	43.320	933	1
5	2,941	43.010	10	43.330	700	2

Obrázek 2.2: Zobrazení hloubky trhu pomocí tabulky. Převzato z http://www.thebull.com.au/.



Obrázek 2.3: Zobrazení hloubky trhu pomocí normálního grafu a grafu s kumulativními součty. Převzato z http://en.wikipedia.org/wiki/Market_depth.

Kapitola 3

Komunikační protokol

Pro účely této práce jsem analyzoval dokumenty zveřejněné NYSE Arca [8] a ISE burzou [5] a v této kapitole vysvětlím, jak funguje posílání zpráv u těchto burz a jakou mají vnitřní strukturu.

Komunikace s burzou probíhá pomocí zpráv, které jsou zasílány v datovém proudu prostřednictvím internetu. Obchodník má možnost si předplatit přístup k těmto informacím a poté může zprávy odebírat. Burzy NYSE Arca a ISE používají několik datových proudů, které jsou rozlišeny podle toho, jaké informace zasílají (např. hloubka trhu, nejlepší nabídka, uskutečněné obchody a další). Pro snížení velikosti přenášených dat a latence se používá FIX/FAST protokol[1], kterým jsou zasílané zprávy komprimovány.

NYSE Arca používá pro zasílání kombinaci UDP a TCP přenosu, přičemž spolehlivý protokol TCP je využíván k získání ztracených paketů z UDP přenosu. ISE posílá zprávy pomocí UDP protokolu.

3.1 Formát zpráv

Každá zpráva zasílaná NYSE Arca a ISE burzou je rozdělena do jednotlivých položek. Význam a počet položek si však definuje každá burza sama ve svých specifikačních dokumentech [8] a [5]. Odtud je také možné vyčíst, jakých hodnot mohou různé položky nabývat. Buď to vyplývá přímo z významu položky, nebo jsou povolené hodnoty uvedeny výčtem, nebo vyplývají z použitého datového typu.

3.1.1 NYSE Arca

Všechny zprávy burzy NYSE Arca vždy začínají stejně formátovanou hlavičkou. Ta obsahuje celkem čtyři položky – délku zprávy, typ zprávy, označení předplatného a časovou známku. Ve zprávách se ještě opakuje pole se sekvenčním číslem (*Message Sequence Number*), které se inkrementuje s každou zaslanou zprávou. Dále se již položky liší podle typu zprávy. Například *Aggregate Quote Message* pak obsahuje položky, které aktualizují hloubku trhu – úroveň, která se má přidat či vymazat, cena, velikost a počet příkazů k nákupu či prodeji na dané úrovni a informace, zda se jedná o stranu nabídky nebo poptávky.

Na obraz burzy neboli hloubku trhu má dále vliv zpráva *System Event Message*, v níž se zasílají kódy událostí, které se mají provést. Pro ilustraci se jedná například o resetování

[1] FAST - FIX Adapted for STreaming - standardizovaný protokol optimalizující přenášení dat po síti finančními institucemi a vytvořený organizací FIX Trading Community

obrazu burzy na straně poptávky nebo nabídky, povolení a zastavení obchodování konkrétních finančních instrumentů aj. Všechny kódy událostí jsou označeny jednopísmenným identifikátorem, přičemž jsou rozlišována velká a malá písmena. V položce *Reset Code* se zasílá indikace pro resetování sekvenčního čísla zpráv.

Další zprávy již přímo neaktualizují hloubku trhu. Jsou to například *Top Quote Message* pro zaslání pouze jedné, a to nejlepší cenové úrovně pro stranu nabídky a poptávky nebo zprávy o proběhlých obchodních transakcích a další.

3.1.2 ISE

Burza ISE nedělí zprávy na hlavičku a tělo, jako je tomu u NYSE Arca. Přesto se ale některé položky opakují ve všech typech zpráv této burzy. Jsou to položky, které definují: typ zprávy, sekvenční číslo, označení datového proudu, přes který je zpráva zasílána, a označení obchodovaného produktu. Zbytek informací je už specifický podle konkrétní zprávy. U zpráv, které aktualizují hloubku trhu (*Depth Snapshot Message a Depth Quote*) je zvláštnost, že počet položek u nich není pevně daný. Pro každou hladinu kurzu se zde vyskytuje sekvence položek a délka zprávy tedy závisí na tom, kolik hladin kurzu má daná zpráva aktualizovat.

Zpráva *Depth Snapshot Message* se posílá v pravidelných intervalech a poskytuje informace o všech úrovních hloubky trhu najednou. Naproti tomu zpráva *Depth Quote* slouží k aktualizaci pouze těch jednotlivých úrovní kurzu, které jsou reálně změněny.

Pro změny v obchodování jednoho finančního instrumentu je zasílána zpráva *Instrument Status* a pro více instrumentů naráz je to *Instrument List Status*.

3.2 Variabilita zpráv

Ve zprávách je rovněž možné zasílat informace o změně obchodování jednotlivých produktů (např. pozastavení obchodování, resetování obrazu pro určitý obchodovaný instrument a podobně). Mnoho z těchto událostí během normální činnosti burzy dlouhou dobu vůbec nemusí nastat, protože nebyly potřeba. Ale programy, které pracují s daty z burzy, musí samozřejmě správně reagovat i na takové zprávy a nesmí dojít k chybě nebo mylné interpretaci. Rovněž změny cenových hladin až k mezním hodnotám se musí testovat. Během reálného provozu obvykle nedochází k velkým cenovým fluktuacím a kurz se nepřiblíží k problematickým krajním hodnotám, a pokud už se tak stane, děje se tak zřídka.

Variabilita zpráv a hodnot, které obsahují, je nízká a důsledkem je, že není přínosné testovat burzovní programy a algoritmy pouze na reálných vzorcích dat. Proto je účelem této práce vytvořit program, který umí generovat zprávy podle zadaných kritérií a zajistit, aby byly vygenerovány všechny možné hodnoty a stavy a případně dále generovat jejich kombinace.

Kapitola 4

Funkční verifikace

Pro účely této práce bylo vhodné využít některých principů funkční verifikace. Informace obsažené v této kapitole jsem čerpal z [12] a [3]. Konkrétní implementace těchto metod je popsána v kapitole 7.

4.1 Možnosti verifikace

Verifikace jako taková je proces ověřování, zda je implementovaný systém korektní, zda se neobjevují nesrovnalosti a systém vyhovuje jisté specifikaci. Existují různé přístupy k verifikaci počítačových programů a systémů. Mezi nejznámější patří formální a funkční verifikace, kde každá má své výhody a nevýhody.

Formální verifikace (*formal verification*) se pomocí matematických metod snaží dokázat správnost ověřovaného systému podle jeho formální specifikace. Ověřuje veškerá možná chování systému a je schopná podat důkaz, že sledovaná vlastnost systému je bezchybná. Její použití v aplikaci Generátor však není vhodné, protože Generátor má testovat programy a systémy pomocí vygenerovaných zpráv. Testování s Generátorem se podobá tzv. *black-box testing*[1] a simulaci, zatímco formální verifice pracuje přímo s testovaným programem nebo jeho zdrojovým kódem, což zde není možné použít.

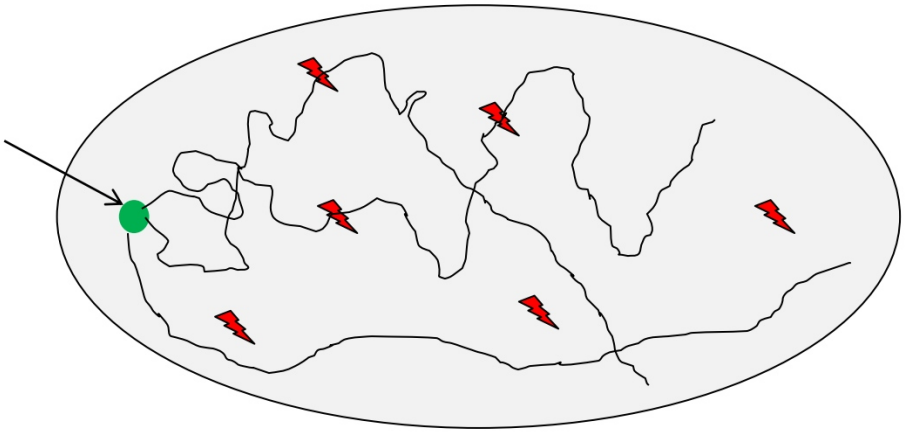
Dalším verifikačním přístupem je funkční verifikace (*functional verification*). Narozdíl od formální verifikace funguje na jiném principu. Ověřuje, zda testovaný systém splňuje specifikaci, a to sledováním jeho vstupů a výstupů v simulaci. To mnohem lépe odpovídá funkci aplikace Generátor. Funkční verifikace se používá pro odhalování chyb hardwarových zařízení. Některé principy a metody by však bylo užitečné použít v programu, protože by díky nim mohl Generátor produkovat data, která by byla pro testování efektivnější.

Nevýhodou funkční verifikace je, že mohou nastat situace, které nemusí být schopna ověřit kvůli přílišné velikosti celého stavového prostoru. Nepodává tudíž důkaz, že celý implementovaný systém je korektní. Přesto dokáže objevit mnoho chyb, které by jinak mohly zůstat skryty. Typicky probíhá tak dlouho, dokud není dosažena určitá úroveň pokrytí, která je považována za dostatečnou.

Procházení stavového prostoru a hledání chyb znázorňuje obrázek 4.1.

Z technik funkční verifikace jsem se rozhodnul využít generování náhodných vstupních vektorů a verifikaci řízenou pokrytím, díky kterým by se nejvíce měla zvýšit efektivita testování.

[1] *Black-box testing* je metoda, která testuje funkcionalitu programu bez znalosti jeho vnitřního uspořádání.



Obrázek 4.1: Pokrývání stavového prostoru funkční verifikací. Převzato ze studijních materiálů předmětu PCS.

4.2 Generování náhodných vstupních vektorů

Technika generování náhodných vstupních vektorů (*constrained-random stimulus generation*) testuje systém proti velkému množství náhodných vstupů (v řádech milionů a více). Korektnost formátu těchto vstupů musí být zajištěna pomocí omezujících podmínek.

Za vstupní vektory se v této práci považují burzovní zprávy a hodnoty v nich, které se po vytvoření Generátorem mají aplikovat na testovaný systém.

Možnost přesně specifikovat, jak mají vygenerovaná data vypadat, je užitečná z více důvodů. Umožní to jednak dodržet specifikaci danou protokolem burzy, ale uživatel programu má navíc možnost změnit podobu generovaných dat podle potřeb testování. Při použití v této práci to umožní ověřit požadovaný rozptyl hodnot a mezních situací.

4.3 Verifikace řízená pokrytím

Verifikace řízená pokrytím (*coverage-driven verification*) modifikuje generování testovacích vstupních vektorů tím způsobem, že vstupní vektory nevytváří úplně náhodně, ale bere v potaz stavy a vlastnosti systému, které již prověřeny byly. U systému můžeme takto sledovat např. pokrytí zdrojového kódu, provedených cest v systému, navštívené stavy konečného automatu apod. Proces pokrývání se průběžně vyhodocuje a je možné po dosažení určitého pokrytí běh programu ukončit. Dále se vytváří statistika o pokrytí, která může uživateli podat informace o průběhu verifikace a o stavech systému, které byly verifikací dostatečně prověřeny.

U této metody se musí jasně určit, co se bude pokrývat. Pro účely této práce by bylo vhodné zavést možnost pokrývat všechny hodnoty, které jsou náhodně generovány. Zprávy zasílané burzou jsou rozdělené do jednotlivých položek a každá položka má ve zprávě jiný význam. Rovněž tak každá položka podle svého typu nabývá jiných rozsahů hodnot. Proto navrhuji cíleně pokrývat hodnoty všech položek ve zprávách, které jsou označeny, že se mají takto náhodně generovat. Celkový stavový prostor by byl dán určením všech možných hodnot, které položky mohou nabývat, a to jednak podle specifikace protokolu dané burzy, ale i podle potřeb testování.

Také zobrazení statistiky o pokrytí je pro uživatele aplikace velmi užitečné, protože si

může ihned udělat přehled, nakolik bylo generování produktivní a zda některé stavy nezůstaly nepokryty. Program by si tedy měl při svém běhu zaznamenávat, které hodnoty vytvořil, a po svém ukončení vypsat zmíněnou statistiku na základě srovnání vygenerovaných hodnot ku celkovému počtu všech možných hodnot.

Kapitola 5

Šablona pro generování

Problémem při testování systémů pro obchodování na finanční elektronické burze je různorodost používaných komunikačních protokolů a přenášených zpráv. Původně bylo zamýšleno vytvořit programový modul pro každou burzu, čímž by se zajistilo, že generátor bude vytvářet zprávy podle specifikací dané burzy. Pokud by však chtěl uživatel přidat další burzu, musel by se vytvořit nový programový modul. Problém by také nastal, jestliže by se změnil formát zasílaných zpráv. Pak by bylo nutné přepracovat stávající modul, který to má na starost.

Aby mohl být generátor zpráv z burzy obecný a mohl být použit pro burzy NYSE Arca a ISE bez jakýchkoliv modifikací v implementaci, bylo namísto zamýšlených programových modulů navrženo použití šablony pro generované zprávy.

Díky šabloně je také možné přesně určit, jaká data se mají generovat, a zúžit nebo naopak rozšířit rozsahy testovaných hodnot. Tím se zároveň definují omezující podmínky pro metodu generování náhodných vstupních vektorů popsanou výše. Šablona rovněž poslouží pro stanovení stavového prostoru, který se má pokrývat metodou verifikace řízené pokrytím. Pokrývají se ty položky, jejichž hodnoty jsou v šabloně označeny, že se mají generovat náhodně z nějaké množiny či rozsahu.

Šablona je popsána jazykem XML. Formát XML byl zvolen z těchto důvodů:

- Formát čitelný strojově i člověkem,
- je sebepopisný,
- je standardizován,
- je nezávislý na platformě.

Šablona se vytvoří na základě protokolu dané burzy a může obsahovat všechny protokolem povolené prvky nebo může být pro účely testování omezena pouze na vybranou množinu dat.

5.1 Požadavky na návrh šablony

Z analýzy specifikačních dokumentů burzy NYSE Arca [8] a ISE [5] jsem určil následující požadavky na to, jak má vypadat šablona. Musí v ní být možné zaznamenat zprávy, které se budou generovat. Množství zpráv není omezené. U zpráv se dá určit pravděpodobnost, s jakou mají být vytvářeny.

Ve zprávě musí být rozlišeny jednotlivé položky. Jejich počet opět není limitován. U položky musí být možno uvést, zda se její vygenerovaná hodnota má interpretovat jako binární data nebo textová. V případě binárních dat je nutné uvést celkovou velikost položky v bajtech. Dále se musí označit, jak se hodnota položky generuje. Je několik možností, jak toto provést. Jednak může být vždy po celou dobu konstantní, nebo je hodnota vybírána z určité množiny stavů. Další možností je generování náhodných hodnot ze specifikovaného intervalu. Intervalů musí být možné zaznamenat více. Pro hodnoty z množiny i pro intervaly musí být také možné určit pravděpodobnost vygenerování.

Poslední možností, jak vygenerovat hodnotu položky, musí být pomocí nějaké uživatelem specifikované funkce. Některé položky ve zprávě (časová známka, sekvenční číslo zprávy) nelze generovat náhodně a nemohou být ani konstantní. Můžou záviset například na předchozí hodnotě nebo obsahují netriviálně generovanou hodnotu. Aby tedy implementace nezávisela na konkrétní burze, může uživatel vytvořit vlastní funkci, která se pro generování takové položky použije, a musí mít možnost ji přidat do programu. Použití této funkce je označeno v šabloně.

Zprávy a jednotlivé položky by mělo být možné pojmenovávat, aby vygenerované zprávy byly srozumitelnější.

5.2 Popis XML šablony

Podle požadavků byla šablona navržena tak, aby umožnila srozumitelně zapsat, jak mají generované zprávy vypadat, protože se předpokládá, že bude vytvářena a upravována uživatelem. Definuje se v ní požadovaný formát zpráv, které se budou generovat, a v každé zprávě jednotlivé položky až na úrovni bajtů. Šablona obsahuje kořenový element `exchange`. V něm se potom uvádějí požadované zprávy pomocí elementu `message`. A konečně pro jednotlivé položky ve zprávě se používá element `field`. Všechny tři elementy je možné pojmenovávat pomocí atributu `name`. Pokud není název určen, doplní se na výchozí hodnoty: *Exchange* pro `exchange`, *MessageX* a *FieldX* pro `message` a `field`, kde *X* je číslo inkrementující se s každou další zprávou či položkou.

Dále je každá položka ve zprávě definovaná elementy `format`, `length`, `type` a `value`. Element `format` se použije k určení, zda se výsledná vygenerovaná data mají interpretovat v binárním nebo textovém formátu. `Length` určuje velikost položky v bajtech a použije se pouze v případě, když je formát binární. `Type` definuje, zda se jedná o konstantu, množinu více hodnot, interval hodnot nebo speciální typ – zmíněnou uživatelem definovanou funkci.

Element `value` pak obsahuje samotnou hodnotu položky. Pokud je položka typu `interval`, jsou ve `value` zanořeny ještě další elementy: `min-value` a `max-value`. Těmi se určí rozsah intervalu. Je možné takto zadat libovolný počet různých intervalů.

Každé zprávě a hodnotě u položky typu `set` a každému intervalu u položky typu `interval` lze stanovit pravděpodobnost, s jakou má dojít k vygenerování. Dělá se tak atributem `probability` a přiřazením hodnoty pravděpodobnosti, což je číslo v rozsahu od nuly do jedné. Pokud se pravděpodobnost jednou uvede, není ji nutné doplňovat u ostatních položek, nýbrž je mezi zbylé rovnoměrně rozpočítána.

Příklad zjednodušené šablony s jedním druhem zprávy a pěti položkami:

```
<exchange name="NYSE">
     <message name="aggregate-quote-msg">
          <field name="message-type">
               <format>char</format>
               <type>constant</type>
               <value>q</value>
          </field>
          <field name="timestamp">
               <format>binary</format>
               <length>4</length>
               <type>special</type>
              <value>nyse_timestamp</value>
          </field>
          <field name="insert-price-level">
               <format>binary</format>
               <length>1</length>
              <type>interval</type>
              <value>
                   <min-value>1</min-value>
                   <max-value>5</max-value>
              </value>
          </field>
          <field name="price">
               <format>binary</format>
               <length>4</length>
               <type>interval</type>
               <value>
                    <min-value>0</min-value>
                    <max-value>300000</max-value>
               </value>
               <value probability="0.8">
                    <min-value>300001</min-value>
                    <max-value>500000</max-value>
               </value>
          </field>
          <field name="side">
               <format>char</format>
               <type>set</type>
               <value>B</value>
               <value>S</value>
          </field>
     </message>
</exchange>
```

Kapitola 6

Volba generátoru náhodných čísel

Aby mohla aplikace Generátor vytvářet náhodné zprávy a kombinace hodnot v nich, potřebuje ke své činnosti generátor náhodných čísel (RNG[1]). Zprávy jsou programem tvořeny v libovolně velkém, resp. neomezeném množství, a pokud by použitý generátor náhodných čísel nebyl kvalitní, mohlo by se to negativně projevit při jeho použití u testování. Proto je vhodné věnovat tomuto tématu zvýšenou pozornost. Špatné generování náhodných čísel může přivodit tyto problémy:

- Neotestování všech hodnot v případě, že generátor čísel není schopen produkovat všechna čísla ze stanoveného rozsahu,
- zvýšení počtu zpráv, které je nutné vytvořit pro stanovené pokrytí, pokud generátor vytváří některá čísla častěji než jiná,
- nedodržení pravděpodobností generování zpráv a hodnot, které jsou zadány uživatelem v XML šabloně.

K získávání náhodných čísel se používají dva přístupy. Jedná se o generování pravých náhodných čísel a generování pseudonáhodných čísel [6].

6.1 Generátor pravých náhodných čísel

Generátor pravých náhodných čísel (TRNG[2]) funguje na principu získávání náhodnosti ze skutečných fyzikálních dějů. TRNG může být implementován jako samostatné zařízení, které odečítá například signál náhodného šumu nebo může využívat radioaktivního rozpadu atomů[3]. Může však rovněž získávat náhodnost přímo od uživatele – měřit prodlevu mezi stisky kláves, zaznamenávat pohyb myši apod.

TRNG jsou preferovány v kryptografii a bezpečnostních aplikacích, kde je důležité zajistit naprostou náhodnost a nepředvídatelnost. Jejich nevýhodou ale je, že kvůli svému designu pracují relativně pomalu, a pokud se vyčerpá nastřádaná zásoba vytvořených náhodných dat, zablokují se a čekají na nová náhodná data. S ohledem na použití v aplikaci Generátor se objevuje další nevýhoda, a tou je nemožnost zreprodukovat stejnou posloupnost náhodných hodnot. To může být užitečné při hledání chyb nebo odlaďování systému, pokud chceme opakovat tvorbu burzovních zpráv se stejnými hodnotami.

[1] RNG - random number generator (generátor náhodných čísel)

[2] TRNG - true random number generator (generátor pravých náhodných čísel)

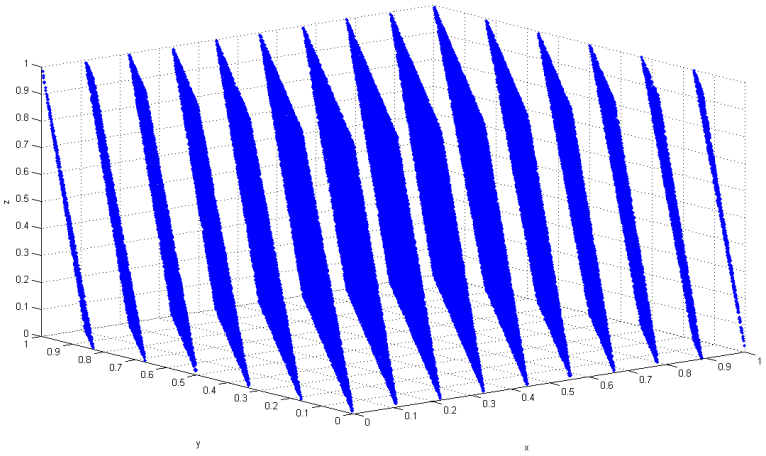
[3] Např. internetová služba HotBits využívá radioaktivního rozpadu izotopu ^{137}Cesia pro generování pravých náhodných čísel.

6.2 Generátor pseudonáhodných čísel

Generátor pseudonáhodných čísel (PRNG[4]) pracuje na rozdíl od TRNG deterministicky – každé následující číslo je vypočítáno algoritmem. PRNG pracují ze své podstaty rychle a jsou neblokující. Před svou činností se musí PRNG nejdříve inicializovat pomocí tzv. semínka (seed). Jestliže použijeme stejné semínko, dostaneme stejnou posloupnost čísel.

Implementací PRNG existuje mnoho druhů a řada z nich trpí vážnými nedostatky. Nejčastějšími jsou příliš krátká perioda generátoru, kdy se sekvence čísel začne brzo opakovat, dále je to problém, kdy generovaná čísla na sobě nejsou nezávislá, ale existuje mezi nimi určitý vztah. Chybou je také nerovnoměrnost rozdělení vytvořených čísel, kdy si lze na velkém vzorku dat všimnout, že některá čísla se opakují častěji než jiná, případně pokud se čísla zaznamenají do vícerozměrného prostoru, tvoří nežádoucí shluky.

Kdysi rozšířený pseudonáhodný generátor RANDU[5] je příkladem velice nekvalitního generátoru. Používal se však díky jeho jednoduché a rychlé implementaci v hardwaru. Obrázek 6.1 demonstruje jednu z jeho slabin – pokud se jím vygenerovaná čísla zaznamenají do vícedimenzionálního prostoru (konktértně třídimenzionálního), nejsou v prostoru rozmístěna rovnoměrně, ale vytvoří shluky 15 rovin, což je způsobeno korelacemi mezi jednotlivými čísly řady [4].



Obrázek 6.1: Čísla z generátoru RANDU zanesená v trojrozměrném prostoru. Převzato z http://en.wikipedia.org/wiki/RANDU#mediaviewer/File:Randu.png.

Zmíněné nedostatky lze minimalizovat použitím kvalitního pseudonáhodného generátoru, který je pro účely aplikace Generátor vhodnější než TRNG. Jelikož je aplikace napsána v jazyce Python 3, bylo by nejjednodušším řešením použít generátor z tohoto jazyka. V následující části se zaměřím na tento generátor, zda je pro účely této práce dostačující.

6.3 PRNG v jazyce Python

Programovací jazyk Python 3 používá jako generátor pseudonáhodných čísel generátor typu Mersenne Twister [1]. Tento generátor je považovan za mnohem kvalitnější než lineární kon-

[4]PRNG - pseudorandom number generator (generátor pseudonáhodných čísel)
[5]lineární kongruentní generátor Parkova-Millerova typu definovaný vztahem $I_{n+1} = (65539 \cdot I_n)\ mod\ 2^{31}$

gruentní generátory a jeho perioda je určena tzv. Mersennovým prvočíslem[6][7]. Nejpoužívanější varianta, kterou využívá rovněž Python, je MT19937 s obrovskou periodou $2^{19937} - 1$. Vygenerovaná čísla vykazují rovnoměrné rozložení až do dimenze 623. V Pythonu je implementace tohoto generátoru optimalizována do modulu v jazyce C, díky čemuž je generování dostatečně rychlé.

Nevýhodou, kvůli které by se tento generátor neměl používat v kryptografii, je, že je teoreticky možné vypočítat čísla, která budou následovat. K tomu musíme mít k dispozici 624 předcházejících vygenerovaných čísel.

Ačkoliv dnes existují PRNG, které jsou v některých vlastnostech lepší než Mersenne Twister (např. generátory typu WELL) [9], pro potřeby v aplikaci Generátor je MT19937 plně dostačující, a proto ho využiji v této práci.

[6]Mersennovy prvočísla jsou prvočísla, která jsou mocninou dvou zmenšenou o jedna. Dají se tedy zapsat vztahem: $M_n = 2^n - 1$

Kapitola 7

Implementace aplikace Generátor

Zadáním této práce bylo vytvořit nástroj, který bude generovat náhodné zprávy podle specifikovaných kritérií. V této kapitole je popsána jeho implementace s použitím věcí popsaných v předchozích kapitolách. Sekce 7.1 se zabývá programovacím jazykem, ve kterém je aplikace vytvořena, sekce 7.2 popisuje vnitřní strukturu programu, parametry programu jsou vysvětleny v sekci 7.3 a poslední sekce 7.4 popisuje, jakým způsobem byly implementovány metody funkční verifikace.

7.1 Volba programovacího jazyka

Pro napsání programu jsem zvolil jazyk Python. Informace k použití tohoto jazyka jsem čerpal z [11] a oficiální webové dokumentace. Tento vysokoúrovňový dynamický programovací jazyk klade důraz na čistotu kódu a dobře čitelný zápis. Jako interpretovaný jazyk nedosahuje takového výkonu jako nativně kompilované jazyky (např. C/C++), avšak díky tomu, že výkonově kritické knihovny jsou napsány v jazyce C, je rychlost stále na dobré úrovni.

Jelikož program nenavazuje na žádné předchozí projekty a nemusí být zpětně kompatibilní s historickým zdrojovým kódem, použil jsem jeho nejnovější verzi 3 (konkrétně 3.2). Tato verze je aktivně vyvíjena a je zde přidávána nová funkcionalita narozdíl od Pythonu 2 (zejména verze 2.7), který je sice rozšířenější, ale již není aktivně vyvíjen, pouze udržován.

Zdrojový kód je napsán tak, že není závislý na konkrétním operačním systému. Aby program Generátor mohl být spuštěn, stačí mít nainstalovaný pouze interpret jazyka Python 3[1].

7.2 Struktura programu

Program je implementován s využitím objektově orientovaného programovacího paradigmatu. Zdrojový kód je rozdělen do logických modulů[2], balíčku[3] a tříd. Následuje popis modulů a balíčku.

[1] Je možné použít i jiné interprety než pouze výchozí a nejpoužívanější CPython – například JIT interpret PyPy nebo IronPython pro prostředí .NET a Mono.

[2] Modulem se v Pythonu rozumí soubor se zdrojovým kódem a příponou `.py` nebo soubor s příponou `.pyc` obsahující přeložený bajtkód.

[3] Balíček je adresář se soubory (moduly) Pythonu a obsahující navíc soubor `__init__.py`

`run.py`

Jedná se o vstupní bod programu, který je jako jediný určen k přímému spouštění. Obsahuje funkci `main()`, ve které se volají metody z ostatních modulů. Také se zde ošetřují výjimky vzniklé při chybných vstupních datech. Program je možné použít v rámci jiného skriptu a jeho úspěšné dokončení ověřit díky návratovému kódu. Přehled návratových kódů je v tabulce 7.1.

Návratový kód	**Událost**
0	Vše v pořádku
1	Chybně zadané parametry příkazové řádky
2	Problém při otevírání souboru se šablonou
3	Soubor se šablonou není validní XML
4	Zjištěny nepovolené hodnoty v souboru se šablonou

Tabulka 7.1: Přehled návratových kódů

`get_params.py`

Ve funkci `get_params()` tohoto stejnojmenného modulu se parsují argumenty příkazové řádky a kontroluje se jejich správné použití. Je využita knihovna `argparse`, která nabízí pokročilou a přitom intuitivní práci s argumenty.

`xml_parser.py`

Modul je umístěn v balíčku `template_parsing` a je v něm pomocí standardní knihovny `xml.etree.ElementTree` parsován a validován soubor XML šablony. ObjektXML stromu se šablonou prochází funkce `validate_template()` a kontroluje, zda se používají pouze přípustné a platné hodnoty. Také naplňuje objekt typu `Template`, a tím vytváří vnitřní reprezentaci šablony.

`parsed_template.py`

Tento modul je součástí balíčku `template_parsing` a obsahuje třídu `Template`, která se použije pro vnitřní reprezentaci šablony. Protože `Template` musí uchovávat i taková data, jako jsou ukazatele na funkce, nebylo možné použít již vytvořený objekt `xml.etree.ElementTree`, protože ten umožňuje uložit v sobě pouze textové literály. Také práce s touto upravenou vnitřní reprezentací šablony přináší vyšší výkon než s použitím `xml.etree.ElementTree`. V případě, že je nastaveno pokrývání stavů a hodnot, obsahuje `Template` také informace o datech, která ještě nebyla pokryta, a informace o již vygenerovaných datech. Ty slouží pro vypočítání statistiky o pokrytí.

`generator.py`

Modul zavádí třídu `Generator`, která se stará o samotnou tvorbu zpráv. To, kolik zpráv se vygeneruje, záleží na zadaných parametrech. Pokud není parametry omezeno, probíhá tak dlouho, dokud není přerušeno uživatelem klávesovou kombinací `Ctrl+C`. Program je totiž během generování připraven zachytit signál *SIGINT*, který uvedená klávesová zkratka vyvolá, zastaví generování zpráv, zobrazí statistiku pokrytí a korektně se ukončí.

Zprávy, které generátor tvoří, jsou objekty typu `Message`.

Pro generování čísel z intervalu používám uniformní rozložení pravděpodobnosti. Pokud by to bylo žádoucí, lze jednoduše doimplementovat i jiná rozložení (Gaussovo, exponenciální atd.).

`message.py`

Třída `Messsage`, která je v tomto modulu definovaná, se chová jako kontejner pro objekt `xml.etree.ElementTree`. Vygenerovaná data jsou tedy uložena v XML struktuře, což zjednodušuje další práci se zprávou, především její tisk na standardní výstup nebo do souboru. Lze si zvolit, zda má být výsledný formát zprávy XML nebo prostý text.

`probability_distribution.py`

Obsahuje funkci pro generování hodnot z intervalu pomocí uniformního rozložení pravděpodobnosti.

`user_functions.py`

Do tohoto modulu může uživatel zadat vlastní funkce, jenž se použijí ke generování hodnot položek, které mají v šabloně typ *special*. Důležitý je název funkce, který se musí shodovat s názvem použitým v šabloně. Každá funkce zde musí vracet nějakou hodnotu.

7.3 Parametry programu

Činnost programu lze modifikovat spuštěním s různými parametry. Formát parametrů musí být v unixovém formátu. Lze libovolně kombinovat krátkou a dlouhou formu[4]. Následuje popis parametrů.

`-m, --max_messages`

Omezení generování na zadaný počet zpráv.

`-s, --max_size`

Generování se ukončí, pokud velikost souboru s vytvořenými zprávami překročí zadanou velikost v bajtech. Je ho možné použít pouze v kombinaci s `--output_file`.

[4]Krátká forma začíná pomlčkou (např. `-m`), dlouhá dvěmi pomlčkami (např. `--cover-all`)

`-c, --cover_all`

Použití tohoto parametru zapíná řízené pokrývání stavů a hodnot definovaných v šabloně. Generování je stále náhodné, ale již jednou vytvořené hodnoty nejsou generovány znovu, pokud ještě zbývají jiné, které je stále ještě třeba pokrýt.

`-t, --template`

Specifikuje, z jakého souboru se načítá XML šablona. Pokud není zadáno, hledá se soubor `template.xml` ve složce `user_templates`.

`-o, --output_file`

Určí, do jakého souboru se mají ukládat zprávy. Když není uvedeno, výstup probíhá na standardní výstup.

`-f, --output_format`

S volbou `xml` mají generované zprávy XML formát, kdežto `plaintext` způsobí formátování jako prostý text.

`-v, --verbose`

S touto možností se průběžně zobrazuje počet vygenerovaných zpráv, který je tištěn na standardní chybový výstup, aby se případně nemíchal s generovanými zprávami.

7.4 Implementace metod funkční verifikace

7.4.1 Generování náhodných vstupních vektorů

Omezující podmínky pro vytváření vstupních testovacích vektorů jsou v XML šabloně zadány pomocí elementů `min-value` a `max-value`, pokud je položka typu `interval`. A pokud je typu `set`, jedná se o elementy `value`. Takto jsou popsány rozsahem nebo výčtem všechny přípustné hodnoty, které se generují náhodně, a je možné je v šabloně libovolně měnit. Zbytek generované zprávy je dále doplněn o konstatní položky a případně o položky typu `special`, které jsou vytvořeny uživatelskou funkcí.

7.4.2 Verifikace řízená pokrytím

Verifikace řízená pokrytím neběží v programu automaticky, ale pouze pokud je Generátor spuštěn s parametrem `--cover_all`. Generování zpráv a hodnot je potom řízeno tak, aby co možná nejrychleji konvergovalo k maximálnímu pokrytí a při dostatečně dlouhém běhu byly pokryty všechny hodnoty a stavy definované v šabloně. Pokrývají se pouze hodnoty položek, které v XML šabloně mají typ `interval` a `set`.

V programu je toto řízené pokrývání implementováno tak, že jsou při startu programu inicializovány množiny všech pokrývaných hodnot. Generování poté probíhá tak, že jsou náhodně voleny pouze hodnoty z těchto množin. Když je hodnota zvolena, je odebrána z množiny a znovu již není generována.

Tento způsob v sobě skýtá nevýhodu, že pokud jsou v šabloně definovány příliš velké rozsahy hodnot (řádově v desítkách milionů), začne být program při spuštění s tímto parametrem paměťově náročný a od určité velikosti stavového prostoru nemusí být spustitelný. Pokud je nutné generovat natolik velké rozsahy hodnot, je lepší tento parametr nepoužít, a tím využít pouze náhodné generování podle specifikace šablony bez řízeného pokrývání.

Parametr `--cover_all` ovšem nijak neovlivňuje generování statistiky o pokrytí, která se vypíše pokaždé při ukončení programu. Statistika poskytuje informace o tom, kolik jakých zpráv bylo vytvořeno, dále informace o pokrytí jednotlivých položek a nakonec celkové dosažené pokrytí všech položek a všech definovaných zpráv.

Kapitola 8

Experimenty s nastavením Generátoru

V této kapitole popíšu experimenty s generováním zpráv, které jsem s vytvořeným programem provedl. Zaměřil jsem se na vliv různých XML šablon a přepínače `--cover_all` na výsledná vygenerovaná data.

Experimenty jsem prováděl na 64bitovém operačním systému Windows 7 s konfigurací Intel Core i5-2430M (2.4GHz, 2 jádra) a 4GB RAM.

Šablony, které jsem pro generování použil, jsou přiloženy ke zdrojovým souborům programu ve složce `user_templates`.

8.1 Malý stavový prostor

Pro tento experiment byla vytvořena šablona `small_state_space.xml`, jejímž základem je zpráva *Depth Quote*, která je zasílána burzou ISE pokaždé, když je aktualizován obraz burzy. Požadované rozsahy dat pro vygenerování byly upraveny tak, aby se zmenšil celkový stavový prostor, který v tomto případě zabírá 167 273 stavů. V praxi se takové nastavení může používat pro testování specifického úzkého rozsahu hodnot a konkrétních vybraných stavů.

Tabulky 8.1 a 8.2 srovnávají generování podle počtu zpráv a podle toho, zda generátor běžel s řízeným pokrýváním stavového prostoru (parametr `--cover_all`) či bez něj. Aby bylo srovnání patrnější, je celkové pokrytí ku počtu zpráv zaneseno do grafu 8.1.

Z tabulek 8.1, 8.2 a především z grafu 8.1 je dobře vidět, jakým způsobem parametr `--cover_all` ovlivňuje generování. U malého počtu zpráv je pokrytí stavů takřka totožné, výsledky se začínají lišit až se zvětšujícím se počtem vygenerovaných zpráv. Pokud Generátor běží v režimu řízeného pokrývání, roste pokrytí přibližně lineárně. Naopak pokud parametr `--cover_all` není použit, je nutné k dosažení stejného pokrytí použít mnohem větší počet zpráv. Lze si také povšimout, že bez tohoto paramteru je obtížné pokrýt stavový prostor ze sta procent – počet zpráv roste exponenciálně se zvyšujícím se pokrytím.

Ze zmíněných tabulek je také patrné, že generování s parametrem `--cover_all` se negativně projevuje na časech potřebných k vyprodukování daného počtu zpráv. To je dáno tím, že aplikace Generátor zajišťuje, aby v tomto režimu nedocházelo k vytváření stejných zpráv a hodnot, které již jednou vytvořeny byly.

Počet zpráv	Celkové pokrytí v (%)	Počet pokrytých stavů	Čas generování (m:s)
100	0,353912	592	0:0,253
500	1,538802	2574	0:0,407
1000	2,779887	4650	0:0,662
5000	9,182594	15360	0:1,987
10000	14,32927	23969	0:4,268
50000	36,943798	61797	0:23,694
100000	55,203171	92340	0:48,258
200000	77,260526	129236	1:11,157
300000	88,483497	148009	1:46,868
500000	96,985766	162231	3:15,354

Tabulka 8.1: Malý stavový prostor, generování bez parametru `--cover_all`.

Počet zpráv	Celkové pokrytí v (%)	Počet pokrytých stavů	Čas generování (m:s)
100	0,367662	615	0:0,606
500	1,686465	2821	0:2,079
1000	3,181027	5321	0:3,925
5000	10,996993	18395	0:18,707
10000	16,982418	28407	0:35,877
50000	41,598465	69583	2:47,604
100000	71,48972	119583	5:26,998
147690	100	167273	7:03,037

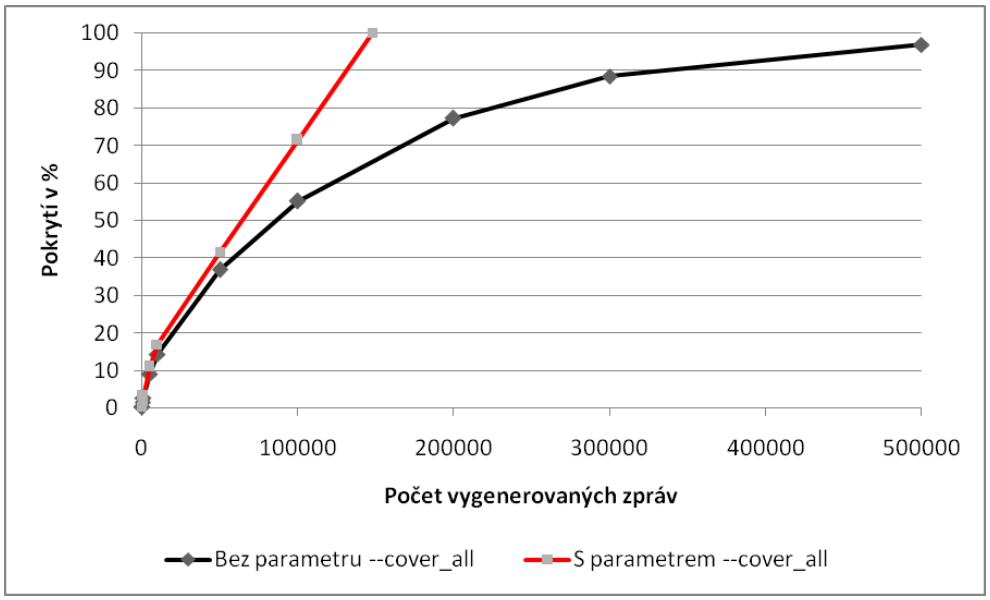
Tabulka 8.2: Malý stavový prostor, generování s parametrem `--cover_all`.

8.2 Středně velký stavový prostor

Pro zjištění, jak se bude program chovat při větším stavovém prostoru a jak budou vypadat výsledné vytvořené zprávy, vznikla XML šablona `middle_state_space.xml`. Ta obsahuje stejně jako v předchozím případě předpis pro zprávu *Depth Quote*, tentokrát však s plnými rozsahy hodnot, které se při obchodování objevují. Stavový prostor zde činí 1 594 540 různých stavů.

Jsou opět uvedeny dvě tabulky 8.3 a 8.4 s výsledky generování s parametrem `--cover_all` a bez něj a dále graf 8.2. Kvůli velkému počtu zpráv nemá x-ová osa grafu (počet vygenerovaných zpráv) lineární měřítko, ale logaritmické. Proto vypadá graf odlišně než u předchozího experimentu.

Z uvedených tabulek a grafu je opět zřejmé, že Generátor podává lepší výsledky s ří-



Obrázek 8.1: Rozdíly v pokrytí s a bez parametru `--cover_all`.

Počet zpráv	**Celkové pokrytí v (%)**	**Počet pokrytých stavů**	**Čas generování (m:s)**
100	0,037126695	592	0:0,264
500	0,154025612	2456	0:0,363
1000	0,290491302	4632	0:0,785
5000	0,814466868	12987	0:2,832
10000	1,522006347	24269	0:5,278
50000	4,392552084	70041	0:26,473
100000	7,047612478	112377	0:51,768
500000	22,21349104	354203	4:47,275
1000000	41,83281699	667041	9:31,017

Tabulka 8.3: Středně velký stavový prostor, generování bez parametru `--cover_all`.

zeným pokrýváním než bez něj. Ovšem na větším stavovém prostoru již rozdíly nejsou tak markantní, ale jsou tím větší, čím více zpráv bylo vygenerováno.

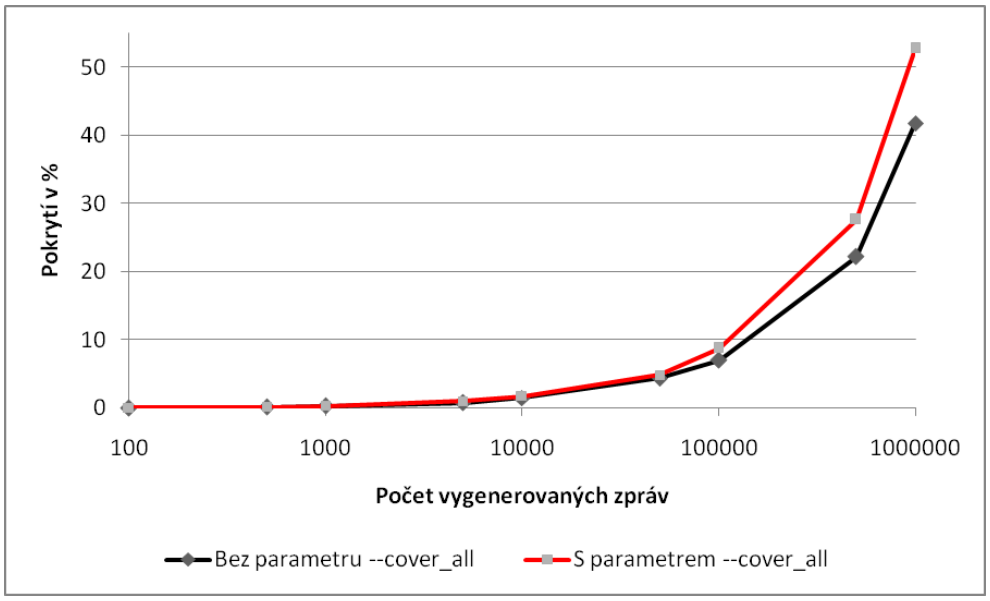
Pokud parametr `--cover_all` není použit, trvá zde vygenerování stejného počtu zpráv zhruba stejně dlouho jako u případu s malým stavovým prostorem. Když však je řízené pokrývání zapnuto, generování je znatelně pomalejší než v předchozím experimentu.

8.3 Shrnutí

Obecně se dá říct, že pro generování zpráv s malým stavovým prostorem je výhodnější použít řízené pokrývání s parametrem `--cover_all`, naopak pro větší stavové prostory

Počet zpráv	Celkové pokrytí v (%)	Počet pokrytých stavů	Čas generování (m:s)
100	0,039196257	625	0:4,587
500	0,176163658	2809	0:18,763
1000	0,333701256	5321	0:34,758
5000	1,028321648	16397	2:47,120
10000	1,781516927	3842	5:31,883
50000	4,88165866	70041	24:26,403
100000	8,753809876	38410	46:30,357
500000	27,65744353	354203	190:03,735
1000000	52,71576756	667041	361:37,452

Tabulka 8.4: Středně velký stavový prostor, generování s parametrem `--cover_all`.



Obrázek 8.2: Rozdíly v pokrytí s a bez parametru `--cover_all`.

použití tohoto parametru příliš zpomaluje generování a jeho výsledky se začnou projevovat až při vygenerování velkého počtu zpráv. Pro opravdu velké stavové prostory pak již řízené pokrývání není vhodné, protože je program příliš časově i paměťově náročný.

Kapitola 9

Analýza reálných dat z burzy

Tato kapitola se zabývá analýzou reálných vzorků dat z burzy a jejich srovnáním s výsledky dosaženými Generátorem.

9.1 Vzorky dat

Byla provedena analýza celkem čtyř vzorků reálných dat z burzy NYSE Arca a burzy ISE. Ve vzorcích se sledovalo, zda obsahují všechny typy zpráv, které burza může zasílat, a jaké je reálné pokrytí vybraných hodnot ve zprávách. Tabulka 9.1 zobrazuje počty všech zpráv v analyzovaných vzorcích. Dále v práci se budu zabývat pouze zprávami, které souvisí s aktualizováním hloubky trhu.

Ve zprávách je sledováno celkové pokrytí nejdůležitějších položek. U zpráv *Depth Quote*, *Depth Snapshot* a *Aqqregate Quote* jsou to položky, které určují úroveň aktualizovaného kurzu, velikosti zadaných příkazů na těchto úrovních a identifikátory instrumentů, ke kterým tyto informace náleží. U zpráv *Instrument Status*, *Instrument List Status* a *System Event* jsou to zase položky, které mění stav obchodování a k nim. Do celkového počtu stavů, ze kterých se vypočítává pokrytí, jsou zahrnuty rozsahy obvyklých hodnot a krajních hodnot.

Vzorek	**Burza**	**Počet zpráv**
Vzorek č. 1	ISE	2 509 514
Vzorek č. 2	ISE	6 071 666
Vzorek č. 3	NYSE Arca	11 383 300
Vzorek č. 4	NYSE Arca	16 368 115

Tabulka 9.1: Počty zpráv ve vzorcích.

9.2 Vzorek č. 1

V tabulce 9.2 jsou uvedeny počty jednotlivých zpráv z burzy ISE, které tento vzorek obsahuje, a dosažené pokrytí.

Vzorek č. 1 v největší míře obsahuje zprávy typu *Depth Quote* a *Depth Snapshot*, které aktualizují velikosti nabídek a poptávek na různých hladinách kurzu. V mnohem menší míře

Typ zprávy	Počet zpráv	Pokrytí v %
Depth Quote	1 319 119	0,06206
Depth Snapshot	183 068	0,09773
Instrument Status	13 560	0,12582
Instrument List Status	1	0

Tabulka 9.2: Počty zpráv ve vzorku č. 1.

jsou zde zastoupeny také zprávy *Instrument Status*, které mění stav obchodování pro jeden konkrétní obchodovaný instrument. Avšak zpráva typu *Instrument List Status*, která mění stav obchodování pro všechny instrumenty, je ve vzorku obsažena pouze jednou.

Co se týče pokrytí, nedosahuje vysokých hodnot, což odpovídá tomu, jak obchodování probíhá. Většina hodnot se ve zprávách stále opakuje a ve vzorku se neobjevují hodnoty z krajů intervalů (především žádné z horních okrajů). Protože se zpráva *Instrument List Status* objevuje pouze jednou, není u ní stanoveno pokrytí z důvodu možného zkreslení.

9.3 Vzorek č. 2

Druhý vzorek z burzy ISE popisuje tabulka 9.3.

Typ zprávy	Počet zpráv	Pokrytí v %
Depth Quote	1 754 733	0,06814
Depth Snapshot	19 455	0,06508
Instrument Status	7 455	0,10798
Instrument List Status	98	1,12071

Tabulka 9.3: Počty zpráv ve vzorku č. 2.

V tomto vzorku jsou jiné poměry počtů zpráv, ale pokrytí se příliš neliší. Objevuje se zde však více zpráv typu *Instrument List Status*, a protože u těchto zpráv není stavový prostor tak velký, mají ze všech zpráv nejvyšší pokrytí.

9.4 Vzorek č. 3

Zprávy ze vzorku č. 3 z burzy NYSE Arca jsou popsány v tabulce 9.4.

Typ zprávy	Počet zpráv	Pokrytí v %
Aqqregate Quote	11 383 300	0,18010
System Event	0	0

Tabulka 9.4: Počty zpráv ve vzorku č. 3.

Z analýzy vyplývá, že v tomto vzorku nejsou žádné zprávy typu *System Event*, proto není ani stanoveno jejich pokrytí. U zpráv *Aqqregate Quote* platí to obdobné, co u jejich protějšků u burzy ISE. Hodnoty se velmi opakují a z krajních intervalů se hodnoty nevyskytují vůbec.

9.5 Vzorek č. 4

Poslední vzorek č. 4 je popsán v tabulce 9.5.

Typ zprávy	**Počet zpráv**	**Pokrytí v %**
Aqqregate Quote	16 368 115	0,19932
System Event	1	0

Tabulka 9.5: Počty zpráv ve vzorku č. 4.

Pokrytí zpráv *Aqqregate Quote* je u vzorku č. 4 o něco vyšší než v předchozím případě, což je způsobeno větším množstvím zpráv ve vzorku. Stále však platí, že pokrytí je značně nízké. Na začátku vzorku se objevuje jedna zpráva *System Event*, která resetuje obchodování. Jedna zpráva však není dostatečně reprezentativní vzorek na určení pokrytí, proto zde není uvedeno.

9.6 Shrnutí

Po analýze všech čtyř vzorků dat je možné konstatovat, že reálné pokrytí hodnot a stavů je podle očekávání velmi nízké. To, že se ve zprávách většina z možných hodnot nikdy nevyskytne, může mít za následek, že chyby v programech, které pracují s daty z burzy, se nemusí po velmi dlouhou dobu vůbec projevit.

S aplikací Generátor je možné právě tyto netypické, ale validní hodnoty generovat, stejně jako vytvářet typy zpráv, které burza zasílá pouze občas.

Kapitola 10

Závěr

Účelem této práce bylo nastudovat problematiku vytváření obrazu burzy (hloubky trhu) a navrhnout a implementovat program, který generuje náhodná data, která obraz burzy mění. Analyzoval jsem zprávy, které aktualizují hloubku trhu burzy NYSE Arca a burzy ISE. Na základě toho jsem zobecnil požadavky, které jsou na vytváření zpráv u těchto burz kladeny, a navrhnul jsem XML šablonu, ve které lze detailně definovat formát a požadované rozsahy testovaných hodnot. V XML šabloně je díky tomu možné popsat i jiné zprávy než pouze ty, které aktualizují obraz burzy.

Diskutoval jsem přínosy použití metod funkční verifikace pro tuto práci. Pro otestování všech stavů, do kterých se burza může dostat, zavádím metodu funkční verifikace řízené pokrytím – sleduje se pokrytí stavů a možných hodnot a generuje se statistika. Samotný obraz burzy a cenové pohyby jsou testovány proti velkému množství náhodně generovaných hodnot za využití metody generování náhodných vstupních vektorů. Hodnoty mohou být definovány v různých intervalech a s různou pravděpodobností generování.

Pozastavil jsem se také u generátoru náhodných čísel, jehož nekvalitní implementace by negativně ovlivnila činnost aplikace a výsledné vygenerované zprávy. Navrhuji použití pseudonáhodného generátoru čísel typu Mersenne Twister ze standardní knihovny jazyka Python, který se vyznačuje vysokou kvalitou generování peudonáhodných čísel.

Implementoval jsem a popsal aplikaci Generátor, která realizuje vytváření zpráv z burzy, a provedl jsem sadu experimentů s různým nastavením a různými XML šablonami. Také jsem provedl porovnání se dvěma vzorky reálných dat z burzy NYSE Arca a rovněž se dvěma vzorky z burzy ISE. Díky tomu jsem zjistil, že tyto zasílané zprávy a hodnoty v nich mají velmi nízkou variabilitu oproti všem možným stavům a hodnotám, které mohou být použity podle specifikačních dokumentů burz.

Program Generátor tedy může sloužit pro testování aplikací, které pracují s daty z burzy, protože umožňuje generovat mezní stavy a široký rozptyl hodnot s vysokou variabilitou podle kriterií daných v XML šabloně a umožňuje tyto hodnoty řízeně pokrývat.

10.1 Další možnosti práce

Pokračování této práce závisí na dalších případných požadavcích, které by byly na aplikaci Generátor kladeny. Může se jednat například o vytvoření XML šablon pro další elektronické burzy. Během jejich vytváření by teoreticky mohl vyvstat problém, že i přes veškerou snahu nebyla XML šablona navržena dostatečně obecně a neumožní tak zaznamenat nějaký specifický typ generovaných dat. Potom by bylo nutné XML šablonu rozšířit. V šabloně by

dále mohlo být užitečné zavést možnost, jak specifikovat, které konkrétní položky se mají pokrývat, pokud Generátor běží s parametrem `--cover_all`.

Také může být vhodné přidat další formáty pro vygenerované zprávy, pokud by to případné specifické testování vyžadovalo. Díky návrhu programu by mělo jít pouze o drobnou změnu ve zdrojovém kódu.

Kapitolou samou pro sebe by mohlo být přímé propojení programu Generátor s testovanými systémy a vytvoření dalších navazujících aplikací, které by za běhu vyhodnocovaly, jak se testované systémy chovají. Na základě toho by se mohly samy upravovat parametry Generátoru přímo během jeho činnosti. Testování programů, které pracují s daty z burzy, by tak mohlo být více automatické a mohlo by běžet libovolně dlouhou dobu.

Literatura

[1] The Python Standard Library - random.py. [online], [cit. 2014-07-10].
URL http://hg.python.org/cpython/file/3.2/Lib/random.py

[2] Dvořák, M.; Kořenek, J.: Low Latency Book Handling in FPGA for High Frequency Trading. 2014: s. 175–178.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=10622

[3] Glasser, M.: *Open Verification Methodology Cookbook*. Springer New York, 2009, ISBN 9781441909688, 236 s.

[4] Hellekalek, P.: Good Random Number Generators Are (Not So) Easy to Find. *Mathematics and Computers in Simulation*, 1998: s. 485–505, ISSN 0378-4754, doi:10.1016/S0378-4754(98)00078-0.

[5] International Securities Exchange, LLC: Market Data Interface (MDI) Programming Manual. [online], 2011-10-27.
URL https://members.ise.com/

[6] Katzgraber, H. G.: Random Numbers in Scientific Computing: An Introduction. *CoRR*, 2010: str. 20, abs/1005.4117.

[7] Matsumoto, M.; Nishimura, T.: Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 1998: s. 3–30, doi:10.1145/272991.272995.

[8] NYSE Euronext: NYSE ArcaBook for Options Client Specification. [online], 2014-02-13.
URL http://www.nyxdata.com/doc/4416

[9] Panneton, F.; L'Ecuyer, P.; Matsumoto, M.: Improved Long-period Generators Based on Linear Recurrences Modulo 2. *ACM Transactions on Mathematical Software*, 2006: s. 1–16, ISSN 0098-3500, doi:10.1145/1132973.1132974.

[10] Pavlát, V.: *Kapitálové trhy a burzy ve světě*. Grada a.s., 1993, ISBN 80-85424-90-8, 393 s.

[11] Pilgrim, M.: *Dive Into Python 3*. Apress, 2009, ISBN 978-1430224150, 360 s.

[12] Piziali, A.: *Functional Verification Coverage Measurement and Analysis*. Springer Publishing Company, Incorporated, 2007, ISBN 9780387739922.

[13] Rejnuš, O.: *Finanční trhy*. Key publishing, 2010, ISBN 978-80-7418-080-4, 660 s.

Příloha A

Obsah CD

Na přiloženém CD je možné nalézt tyto adresáře a soubory:

Soubor nebo adresář	Popis
text/	Tato písemná zpráva ve formátu PDF a její zdrojové soubory v LaTeXu.
text/fig/	Obrázky použité v této písemné zprávě.
src/	Zdrojové soubory programu Generator v jazyce Python.
install/	Instalační soubor jazyka Python 3.2 pro operační systémy Linux, Windows a Mac OS X.
README.txt	Textový soubor s návodem k použití programu.